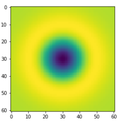
Name : W. K. D. D. Senuruk
Index Number : 190589H

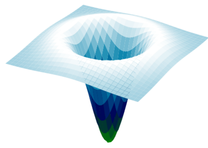
EN2550 Exercise 5 on Blobs and Fitting Basics

(1)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv

sigma  = 10
hw = 3 * sigma
X, Y = np.meshgrid(np.arange(-hw, hw + 1, 1), np.arange(-hw, hw + 1, 1))
log = 1 / (2 * np.pi * sigma**2)*(X**2/(sigma**2) + Y**2/(sigma**2) - 2)* np.exp(-(X**2 + Y**2)/(2 * sigma**2))
plt.imshow(log)
plt.show()
```

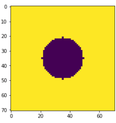


In [2]:
```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter

fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection="3d")

surf = ax.plot_surface(X, Y, log, cmap=cm.ocean, linewidth = 0, antialiased=True)

ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter("%.02f"))
plt.axis("off")
plt.show()
```



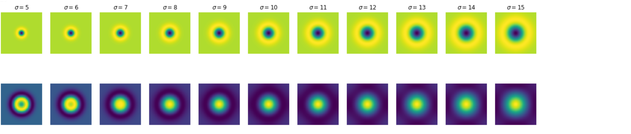
(2)

In [3]:
```
w, h = 71, 71
hw, hh= w//2, h//2

f = np.ones((h, w), dtype=np.float32)*255
X, Y =np.meshgrid(np.arange(-hh, hh + 1, 1), np.arange(-hw, hw + 1, 1))
r = w//5
f *= X**2 + Y**2 > r**2
plt.imshow(f)
plt.show()
```



In [4]:
```
s = 11
fig, ax = plt.subplots(2, s, figsize=(20, 5))
scale_space = np.empty((h, w, s), dtype=np.float32)
sigmas = np.arange(5, 16, 1)
for i, sigma in enumerate(sigmas):
    log_hw = 3 * np.max(sigmas)
    X, Y = np.meshgrid(np.arange(-log_hw, log_hw + 1, 1), np.arange(-log_hw, log_hw + 1, 1))
    log = 1 / (2 * np.pi * sigma**2)*(X**2/(sigma**2) + Y**2/(sigma**2) - 2)* np.exp(-(X**2 + Y**2)/(2 * sigma**2))
    f_log = cv.filter2D(f, -1, log)
    scale_space[:, :, i] = f_log
    ax[0, i].imshow(log)
    ax[0, i].axis("off")
    ax[0, i].set_title(r"$\sigma = {}$".format(sigma))
    ax[1, i].imshow(f_log)
    ax[1, i].axis("off")

indices = np.unravel_index(np.argmax(scale_space, axis=None), scale_space.shape)
print(indices)
print(sigmas[indices[2]])
```

```
(35, 35, 5)
10
```



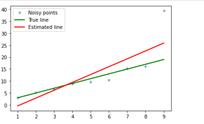
We have find the extremum in the x axis, y axis and the scale axis. So, the characteristic scale of this image is sigma = 10, which corresponds to a circle of radius 14.

(3)

In [5]:
```
img1 = cv.imread(r"Images/gim/img1.ppm", cv.IMREAD_GRAYSCALE)
img2 = cv.imread(r"Images/gim/img3.ppm", cv.IMREAD_GRAYSCALE)

sift = cv.SIFT_create()

kp1, des1 = sift.detectAndCompute(img1,None)
kp2, des2 = sift.detectAndCompute(img2,None)

bf = cv.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)

good = []
for m, n in matches:
    if (m.distance < 0.5 * n.distance):
        good.append([m])

plt.subplots(figsize=(18, 9))
img3 = cv.drawMatchesKnn(img1, kp1, img2, kp2, good, None, matchColor=(0, 255, 0), flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
plt.axis("off")
plt.imshow(img3),plt.show()
plt.show()
```

(4)

In [6]:
```
m = 2
c = 1
x = np.arange(1, 10, 1)
np.random.seed(45)
sigma = 1
n = sigma * np.random.randn(len(x))
o = np.zeros(x.shape)
o[-1] = 20
y = m * x + c + n + o

n = len(x)
X = np.concatenate([x.reshape(n, 1), np.ones((n, 1))], axis=1)
B = np.linalg.pinv(X.T @ X) @ X.T @ y
mstar = B[0]
cstar = B[1]

plt.plot(x, y, '+', label='Noisy points')
plt.plot([x[0], x[-1]], [m*x[0] + c, m*x[-1] + c], color='g', linewidth=2, label='True line')
plt.plot([x[0], x[-1]], [mstar*x[0] + cstar, mstar*x[-1] + cstar], color='r', linewidth=2, label='Estimated line')
plt.legend()
plt.show()
```



(5)

In [7]:
```
m = 2
c = 1
x = np.arange(1, 10, 1)
np.random.seed(45)
sigma = 1
n = sigma * np.random.randn(len(x))
o = np.zeros(x.shape)
# o[-1] = 20
y = m * x + c + n + o

n = len(x)

u11 = np.sum((x - np.mean(x))**2)
u12 = np.sum((x - np.mean(x))*(y - np.mean(y)))
u21 = u12
u22 = np.sum((y - np.mean(y))**2)

U = np.array([[u11, u12], [u21, u22]])
W, V = np.linalg.eig(U)
ev_corresponding_to_smallest_ev = V[:, np.argmin(W)]

a = ev_corresponding_to_smallest_ev[0]
b = ev_corresponding_to_smallest_ev[1]
d = a*np.mean(x) + b*np.mean(y)

mstar = -a/b
cstar = d/b

plt.plot([x[0], x[-1]], [m*x[0] + c, m*x[-1] + c], color='g', linewidth=2, label='True line')
plt.plot([x[0], x[-1]], [mstar*x[0] + cstar, mstar*x[-1] + cstar], color='r', linewidth=2, label='Estimated line')
plt.plot(x, y, '+', label='Noisy points')
plt.legend(loc='best')
plt.show()
```